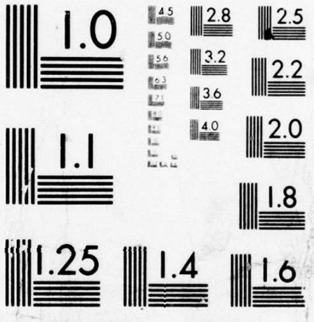
AD-A069 451   NAVAL RESEARCH LAB WASHINGTON D C   F/G 12/1
COMPUTER-AIDED DISCOVERY OF A FAST MATRIX-MULTIPLICATION ALGORI--ETC(U)
MAY 79   R W JOHNSON, A M MCLOUGHLIN

UNCLASSIFIED   NRL-MR-3994   NL

1 OF 1
AD
A069451

END
DATE
FILMED
7--79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

NRL Memorandum Report 3994

Computer-Aided Discovery of a Fast Matrix-Multiplication Algorithm

RODNEY W. JOHNSON

Information Systems Staff
Communications Sciences Division

and

AILEEN M. MCLOUGHLIN

Trinity College, Dublin, Ireland

May 7, 1979

NAVAL RESEARCH LABORATORY
Washington, D.C.

Approved for public release, distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER NRL Memorandum Report 3994	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 COMPUTER-AIDED DISCOVERY OF A FAST MATRIX-MULTIPLICATION ALGORITHM.		5. TYPE OF REPORT & PERIOD COVERED 9 Final report on an NRL problem.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 Rodney W. Johnson ~~and~~ Aileen M. McLoughlin		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, DC 20375 12 13p.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NRL Problem B02-35 61153N, RR014-09-41
11. CONTROLLING OFFICE NAME AND ADDRESS 14 NRL-MR-3994		12. REPORT DATE May 7, 1979
		13. NUMBER OF PAGES 12
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 11 7 May 79		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

16 RR01409 17 RR0140941

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Matrix multiplication
Strassen's algorithm
Computational complexity

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computer program was written that searches for fast matrix-multiplication algorithms by seeking roots of a certain multivariate polynomial. An algorithm was discovered that, like the one discovered by Laderman, uses 23 noncommutative multiplications in multiplying 3-by-3 matrices. The new algorithm is demonstrably inequivalent to Laderman's in a sense that is made precise.

251 950

CONTENTS

COMPUTER-AIDED DISCOVERY OF A FAST MATRIX-MULTIPLICATION ALGORITHM

I. INTRODUCTION

Multiplying two n-by-n matrices by straightforward evaluation of the usual definition,

$$Z_{ik} = \sum_{j=1}^{n} X_{ij} Y_{jk} ,$$

involves multiplying n^3 pairs of numbers and performing a proportionate number of other elementary operations, such as additions; the total number of operations is $O(n^3)$ as n increases. A celebrated algorithm of Strassen's [1] requires only $O(n^a)$ operations, where the exponent a is $\log_2 7$, or about 2.807. Strassen's is one of a class of similar algorithms. Each algorithm of the class is based on a method for reducing the problem of multiplying two n-by-n matrices to that of multiplying M pairs of $\lceil n/N \rceil$-by-$\lceil n/N \rceil$ matrices,* where n is arbitrary and M and N are fixed integers characteristic of the algorithm. The total number of operations used by the algorithm is $O(n^a)$, where $a = \log_N M$ (provided $\log_N M > 2$).

For Strassen, N = 2 and M = 7. Winograd [2] has shown that when N = 2, the best attainable M is 7.

An algorithm due to Laderman [3] has N = 3 and M = 23. With N = 3, it is an open question whether smaller values of M are attainable; improvement over the best known value of a would require $M \leq 21$. When N = 4, Strassen's algorithm achieves M = 49, with $M \leq 48$ needed for improvement. With N = 5, Schachtel [4] has given an algorithm with M = 103, and $M \leq 89$ is needed for improvement over known results. Strassen's result has so far been surpassed only by Pan [5], who recently described a family of algorithms one of which has N = 70 and M = 143640. The corresponding exponent a is $\log_{70} 143640$, or about 2.795.

*The upper half-brackets denote the "ceiling function" — the least integer not less than n/N.

Note: Manuscript submitted March 12, 1979.

The algorithms mentioned appear to be products of unaided human ingenuity; Laderman, at least, explicitly denies having used a computer in obtaining his result. We report here some results of using a computer to search for such matrix-multiplication schemes. We wrote a short APL version of the proposed search procedure to gain some experience before deciding whether to devote substantial effort to writing a more efficient version; we set ourselves the goal of reproducing the known results for N = 2 and 3. The search with N = 2 and M = 7 was successful; the algorithm discovered is equivalent to Strassen's in a sense that will be made explicit further down. The search with N = 3 and M = 23 neither failed nor rediscovered Laderman's algorithm; it turned up an algorithm that, in the sense mentioned, is inequivalent to Laderman's. This algorithm lacks certain desirable properties that Laderman's has, but is presented here for the sake of any clues it may offer to the structure of the class of algorithms it belongs to. We have not yet improved on previously known values of N and M.

In the next section, partly to establish some notation, we give a brief background discussion of the form of the algorithms we are considering. In the third section we make explicit, as promised, a notion of equivalence of two such algorithms. In the fourth section, we describe the search procedure, and in the fifth we present the algorithm discovered.

II. FORM OF THE ALGORITHMS

Each of the algorithms uses a scheme for multiplying N-by-N matrices that is of the form

$$(1) \qquad Z_{nm} = \sum_{r=1}^{M} C_{mn}^{(r)} \left(\sum_{i,j=1}^{N} A_{ij}^{(r)} X_{ij} \right) \left(\sum_{k,l=1}^{N} B_{kl}^{(r)} Y_{kl} \right) ,$$

where $A^{(r)}$, $B^{(r)}$, and $C^{(r)}$ are fixed N-by-N matrices of real numbers. Such a scheme does not depend on commutativity of the elements X_{ij} and Y_{kl} of the matrices being multiplied--it works even when X_{ij} and Y_{kl} belong to some noncommutative algebra over the real numbers. In particular, X_{ij} and Y_{kl} may be matrices: if X and Y are n-by-n matrices of real numbers, and n is a multiple of N, then we may, by partitioning, regard X and Y as N-by-N matrices whose elements X_{ij} and Y_{kl} are (n/N)-by-(n/N) matrices. If n is not originally a multiple of N, we may pad X and Y with rows and columns of zeros until their size becomes a multiple of N. In any case, (1) gives us a method for computing the product Z of X and Y by multiplying M pairs of smaller matrices, of the size of X_{ij} and Y_{kl}. We compute each of the products of smaller matrices by applying the same method recursively; ultimately the problem reduces to one of multiplying 1-by-1 matrices.

Besides the M multiplications of pairs of ("smaller") matrices, (1) involves several multiplications of matrices by scalar coefficients $A_{ij}^{(r)}$, $B_{kl}^{(r)}$, and $C_{mn}^{(r)}$. For Strassen's, Laderman's, and Schachtel's algorithms, but not for the one we will present here, the scalar coefficients are all either 0, +1, or -1, and the corresponding multiplications consequently become trivial. This simple form for the coefficients reduces the cost of an algorithm by a considerable constant factor and is therefore important practically; however, the asymptotic exponent a is not affected: in the bound $O(n^a)$ on the cost of the algorithm, we still have $a = \log_N M$ whether the coefficients are 0's, 1's, and -1's or are arbitrary floating-point numbers.

III. EQUIVALENCE

A necessary and sufficient condition for (1) to define Z as the matrix product of X and Y, as opposed to some other bilinear function, is that

$$\sum_{r=1}^{M} A_{ij}^{(r)} B_{kl}^{(r)} C_{mn}^{(r)} = \delta_{ni} \delta_{jk} \delta_{lm} . \tag{2}$$

A number of simple transformations on the families A, B, and C of coefficients carry solutions of (2) into other solutions of (2). Such transformations may be considered as elementary equivalences between the matrix-product algorithms corresponding to the families of coefficients. Two of the simplest are the replacement

$$A^{(r)}, B^{(r)}, C^{(r)} \rightarrow A^{(r')}, B^{(r')}, C^{(r')} , \tag{3}$$

for some permutation r → r' of the indices 1,...,M, and cyclic permutation of A, B, C:

$$A, B, C \rightarrow C, A, B . \tag{4}$$

A third such transformation is transposition together with reversal of the order of A, B, C (we write $\tilde{A}^{(r)}$ for the transpose of $A^{(r)}$):

$$A^{(r)}, B^{(r)}, C^{(r)} \rightarrow \tilde{C}^{(r)}, \tilde{B}^{(r)}, \tilde{A}^{(r)} , \tag{5}$$

A fourth is to choose real numbers a_r, b_r, and c_r such that $a_r b_r c_r = 1$ for $r = 1,\dots,M$, and to map

$$(6) \qquad A^{(r)},\ B^{(r)},\ C^{(r)} \rightarrow a_r A^{(r)},\ b_r B^{(r)},\ c_r C^{(r)} \ .$$

The fifth and last such transformation we will list is to choose three nonsingular N-by-N matrices P, Q, and R and make the replacement

$$(7) \qquad A^{(r)},\ B^{(r)},\ C^{(r)} \rightarrow QA^{(r)}R^{-1},\ RB^{(r)}P^{-1},\ PC^{(r)}Q^{-1} \ .$$

We will call two solutions of (2), or the corresponding algorithms, equivalent if one can be turned into the other by a combination of transformations of the types (3)--(7).

To illustrate the fifth type of transformation, we display the coefficients of Strassen's original algorithm [1] (Table 1) and those of a version due to Winograd [6], which uses the same number of multiplications but fewer additions (Table 2). Strassen's algorithm is

Table 1
Coefficients for
Strassen's Algorithm

r	$A^{(r)}$	$B^{(r)}$	$C^{(r)}$
1	$\begin{matrix}1 & 0\\0 & 1\end{matrix}$	$\begin{matrix}1 & 0\\0 & 1\end{matrix}$	$\begin{matrix}1 & 0\\0 & 1\end{matrix}$
2	$\begin{matrix}0 & 0\\1 & 1\end{matrix}$	$\begin{matrix}1 & 0\\0 & 0\end{matrix}$	$\begin{matrix}0 & 1\\0 & -1\end{matrix}$
3	$\begin{matrix}1 & 0\\0 & 0\end{matrix}$	$\begin{matrix}0 & 1\\0 & -1\end{matrix}$	$\begin{matrix}0 & 0\\1 & 1\end{matrix}$
4	$\begin{matrix}0 & 0\\0 & 1\end{matrix}$	$\begin{matrix}-1 & 0\\1 & 0\end{matrix}$	$\begin{matrix}1 & 1\\0 & 0\end{matrix}$
5	$\begin{matrix}1 & 1\\0 & 0\end{matrix}$	$\begin{matrix}0 & 0\\0 & 1\end{matrix}$	$\begin{matrix}-1 & 0\\1 & 0\end{matrix}$
6	$\begin{matrix}-1 & 0\\1 & 0\end{matrix}$	$\begin{matrix}1 & 1\\0 & 0\end{matrix}$	$\begin{matrix}0 & 0\\0 & 1\end{matrix}$
7	$\begin{matrix}0 & 1\\0 & -1\end{matrix}$	$\begin{matrix}0 & 0\\1 & 1\end{matrix}$	$\begin{matrix}1 & 0\\0 & 0\end{matrix}$

Table 2
Coefficients for
Winograd's Algorithm

r	$A^{(r)}$	$B^{(r)}$	$C^{(r)}$
1	$\begin{matrix}-1 & 0\\1 & 1\end{matrix}$	$\begin{matrix}1 & -1\\0 & 1\end{matrix}$	$\begin{matrix}0 & 1\\1 & 1\end{matrix}$
2	$\begin{matrix}0 & 0\\0 & 1\end{matrix}$	$\begin{matrix}1 & -1\\-1 & 1\end{matrix}$	$\begin{matrix}0 & -1\\0 & 0\end{matrix}$
3	$\begin{matrix}-1 & 0\\0 & 0\end{matrix}$	$\begin{matrix}-1 & 0\\0 & 0\end{matrix}$	$\begin{matrix}1 & 1\\1 & 1\end{matrix}$
4	$\begin{matrix}0 & 0\\1 & 1\end{matrix}$	$\begin{matrix}-1 & 1\\0 & 0\end{matrix}$	$\begin{matrix}0 & 0\\1 & 1\end{matrix}$
5	$\begin{matrix}0 & 1\\0 & 0\end{matrix}$	$\begin{matrix}0 & 0\\1 & 0\end{matrix}$	$\begin{matrix}1 & 0\\0 & 0\end{matrix}$
6	$\begin{matrix}1 & 0\\-1 & 0\end{matrix}$	$\begin{matrix}0 & -1\\0 & 1\end{matrix}$	$\begin{matrix}0 & 1\\0 & 1\end{matrix}$
7	$\begin{matrix}1 & 1\\-1 & -1\end{matrix}$	$\begin{matrix}0 & 0\\0 & 1\end{matrix}$	$\begin{matrix}0 & 0\\1 & 0\end{matrix}$

transformed by (7) into Winograd's if we set

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix} .$$

The two algorithms are thus equivalent in the sense we have defined.

IV. SEARCH PROCEDURE

Solutions of (2) correspond to zeros of

$$\text{(8)} \qquad \left(\sum_{r=1}^{M} A_{ij}^{(r)} B_{kl}^{(r)} C_{mn}^{(r)} - \delta_{ni}\delta_{jk}\delta_{lm} \right)^2 ,$$

which is a nonnegative function of the A's, the B's, and the C's. We sought solutions of (2) by trying to minimize (8). Although (8) is a sixth-degree polynomial, it is only quadratic in the A's when the B's and C's are held fixed; likewise it is quadratic as a function of the B's alone or of the C's alone. The APL program minimizes (8) with respect to the C's while holding the A's and B's fixed, then minimizes with respect to the B's with fixed A's and C's, and continues thus cyclically. The reason for so constructing the program was mainly programming convenience. One of the APL primitive functions, written as ⌹, produces solutions to sets of linear equations, including least-squares solutions to overdetermined sets. It is quite straightforward to express in terms of this function the solution to quadratic minimization problems such as minimizing (8) with respect to the A's. In addition to the cyclic program just described, a simple straight-line search program was written. The two programs used in alternation frequently proved to be more effective than either used alone.

One disadvantage to seeking solutions of (2) by minimizing (8) is that negative results are inconclusive: if the computation happens to converge to a nonzero local minimum of (8), that is no proof that (8) does not have a zero elsewhere. Another difficulty was more troublesome in practice than nonzero local minima: "zeros at infinity." It is possible for certain of the A's, B's, and C's to tend to infinity in such a way that (8) tends to zero. This difficulty was countered with a modification of the expression the programs were attempting to minimize; a term

$$\epsilon \sum_{rij} \left((A_{ij}^{(r)})^2 + (B_{ij}^{(r)})^2 + (C_{ij}^{(r)})^2 \right)$$

was added to (8). The coefficient ϵ was adjusted by trial and error, interactively, so that, if possible, the magnitudes of the A's, B's, and C's would stay bounded or decrease at the same time that the value of (8) was decreasing. If a suitable value for ϵ could not be found, new random starting values were chosen for the A's, B's, and C's, and the search was begun again.

The procedure just described is unlikely to lead to a solution of (2) in small integers, even if one exists; with any integer solution, transformations (6) and (7) associate a whole family of equivalent solutions, most of which do not consist of integers. Functions for performing transformations of the forms (6) and (7) were written. When the minimization procedure appeared to be converging to a zero of (8), these functions were used in an attempt to assure that the solution would be expressible in a simple form--if possible, in terms of 1's, 0's, and -1's.

V. THE NEW ALGORITHM

The solution we obtained, after simplification, is shown in Table 3. We have not succeeded in transforming the solution to a form consisting entirely of small integers: there remain several rational numbers with 2's and 3's in their numerators and denominators. In this respect, and in general lack of symmetry, this solution compares distinctly unfavorably with the coefficients of Laderman's algorithm, which are given in Table 4. The algorithm resulting from the new solution does, however, have the same exponent $a = \log_3 23$ as Laderman's, and it is provably inequivalent to Laderman's.

To prove inequivalence, we point out that, except for permutations, the transformations (3)--(7) leave the ranks of the matrices $A^{(r)}$, $B^{(r)}$, and $C^{(r)}$ unchanged. All the matrices in Table 3 have rank 1 or 2. But six of the matrices in Table 4 ($A^{(1)}$, for instance) have rank 3. Therefore, no combination of transformations (3)--(7) can change the solution in Table 4 into that in Table 3. That is, the two algorithms with coefficients in Tables 3 and 4 are inequivalent in the sense we have defined.

Table 3--Coefficients for New Algorithm

r	$A^{(r)}$			$B^{(r)}$			$C^{(r)}$		
	1	0	-1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
2	1	1	-1	1	1	0	1	1	0
	0	0	0	0	0	0	0	0	0
	1	-1	-1	0	1	-1	0	0	0
3	-1	1	1	0	0	0	1	0	0
	1/3	1	1	0	0	0	0	0	0
	0	1	0	0	0	0	1	1	0
4	0	0	0	1	0	0	-1	-1	0
	0	0	0	0	0	0	0	0	0
	0	-1	0	0	-1	1	0	0	1
5	0	1	0	-1	-1	0	1	0	-1
	1/3	1	0	0	0	0	0	0	2
	0	0	1	0	1	-1	0	1	0
6	0	0	-1	0	0	0	0	-1	0
	0	0	-1	0	1	1	3/2	3/2	0
	0	0	0	0	0	0	0	0	2
7	0	0	0	0	1	1/2	0	0	-2
	0	-1	1	0	0	0	0	0	2
	1	0	0	0	0	-1	0	0	0
8	-1	0	0	0	0	0	-1	0	0
	0	0	0	0	0	0	-1	0	0
	0	0	1	1	0	0	1	1	0
9	0	0	0	0	0	0	-1	-1	0
	0	0	0	1	0	0	1	1	0
	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	-1	0	0	0
	0	0	0	0	0	0	-1	-1	0
	0	0	0	-1	-1/3	1/3	0	3/2	-1
11	-1	0	1	-2/3	-2/3	0	-3/2	-3/2	0
	0	0	1	-1	-1	0	0	0	0
	0	0	0	0	0	0	0	0	-2
12	0	0	0	1	1	0	-1	0	1
	-1/3	-1	1	0	0	0	0	0	-2

Table 3 (continued)--Coefficients for New Algorithm

r	$A^{(r)}$			$B^{(r)}$			$C^{(r)}$		
	0	0	-2/3	1	1	-1	0	3/2	0
13	-1/3	0	1	0	0	0	-3/2	-3/2	0
	0	0	1	1	1	0	3	3	0
	0	0	0	0	0	0	0	-1	1
14	1	0	-1	1	1	0	0	0	0
	0	0	0	1	1	0	0	0	0
	0	0	-1/2	1	1/2	-1/2	0	0	0
15	0	0	3/2	0	0	0	0	0	0
	0	0	3/2	1	1/2	-1/2	-2	-2	0
	0	0	0	-1	-1/3	1/3	0	0	-1
16	1	0	-1	1/3	1/3	0	0	0	0
	1	0	-1	0	0	0	0	0	0
	0	-1	0	0	0	0	0	-1	-1
17	0	1	0	-1	0	1/2	0	1	1
	0	0	1	0	0	-1/2	0	0	-2
	0	0	0	-1	-1	0	0	0	0
18	1	0	0	0	0	0	-1	-1	0
	0	0	0	-1	-1	0	-1	-1	0
	0	-1	0	0	0	0	0	-1/2	1/2
19	0	1	0	0	0	-1	0	1/2	-1/2
	0	0	0	0	0	-1	0	-1	1
	0	0	0	0	0	0	0	1	-1
20	0	0	0	0	-1	-1/2	0	-1	1
	0	0	-1	0	-1	-1/2	0	0	0
	0	0	0	0	-1	-1/2	0	0	0
21	0	0	0	0	0	0	0	0	-2/3
	1	0	0	0	0	0	0	0	-2/3
	0	1	-1	0	0	0	0	-1	0
22	0	-1	1	0	0	0	0	1	0
	0	0	0	0	0	-1	0	-1	0
	0	1	0	0	-1	1	0	0	1
23	0	-1	0	0	-1	-1	0	0	-1
	0	-1	0	0	0	0	0	0	2

Table 4--Coefficients for Laderman's Algorithm

r	$A^{(r)}$	$B^{(r)}$	$C^{(r)}$	r	$A^{(r)}$	$B^{(r)}$	$C^{(r)}$
1	1 1 1	0 0 0	0 0 0	13	0 0 1	0 0 0	0 0 1
	-1 -1 0	0 1 0	1 0 0		0 0 0	0 1 0	0 0 1
	0 -1 -1	0 0 0	0 0 0		0 0 -1	0 -1 0	0 0 0
2	1 0 0	0 -1 0	0 1 0	14	0 0 1	0 0 0	1 1 1
	-1 0 0	0 1 0	0 1 0		0 0 0	0 0 0	1 0 1
	0 0 0	0 0 0	0 0 0		0 0 0	1 0 0	1 1 0
3	0 0 0	-1 1 0	0 1 0	15	0 0 0	0 0 0	0 0 0
	0 1 0	1 -1 -1	0 0 0		0 0 0	0 0 0	1 0 1
	0 0 0	-1 0 1	0 0 0		0 1 1	-1 1 0	0 0 0
4	-1 0 0	1 -1 0	0 1 0	16	0 0 -1	0 0 0	0 1 0
	1 1 0	0 1 0	1 1 0		0 1 1	0 0 1	0 0 0
	0 0 0	0 0 0	0 0 0		0 0 0	1 0 -1	1 1 0
5	0 0 0	-1 1 0	0 0 0	17	0 0 1	0 0 0	0 1 0
	1 1 0	0 0 0	1 1 0		0 0 -1	0 0 1	0 0 0
	0 0 0	0 0 0	0 0 0		0 0 0	0 0 -1	0 1 0
6	1 0 0	1 0 0	1 1 1	18	0 0 0	0 0 0	0 0 0
	0 0 0	0 0 0	1 1 0		0 1 1	0 0 0	0 0 0
	0 0 0	0 0 0	1 0 1		0 0 0	-1 0 1	1 1 0
7	-1 0 0	1 0 -1	0 0 1	19	0 1 0	0 0 0	1 0 0
	0 0 0	0 0 1	0 0 0		0 0 0	1 0 0	0 0 0
	1 1 0	0 0 0	1 0 1		0 0 0	0 0 0	0 0 0
8	-1 0 0	0 0 1	0 0 1	20	0 0 0	0 0 0	0 0 0
	0 0 0	0 0 -1	0 0 0		0 0 1	0 0 0	0 1 0
	1 0 0	0 0 0	0 0 1		0 0 0	0 1 0	0 0 0
9	0 0 0	-1 0 1	0 0 0	21	0 0 0	0 0 1	0 0 0
	0 0 0	0 0 0	0 0 0		1 0 0	0 0 0	0 0 0
	1 1 0	0 0 0	1 0 1		0 0 0	0 0 0	0 1 0
10	1 1 1	0 0 0	0 0 0	22	0 0 0	0 1 0	0 0 0
	0 -1 -1	0 0 1	0 0 0		0 0 0	0 0 0	0 0 1
	-1 -1 0	0 0 0	1 0 0		1 0 0	0 0 0	0 0 0
11	0 0 0	-1 0 1	0 0 1	23	0 0 0	0 0 0	0 0 0
	0 0 0	1 -1 -1	0 0 0		0 0 0	0 0 0	0 0 0
	0 1 0	-1 1 0	0 0 0		0 0 1	0 0 1	0 0 1
12	0 0 -1	0 0 0	0 0 1				
	0 0 0	0 1 0	1 0 1				
	0 1 1	1 -1 0	0 0 0				

REFERENCES

1. V. Strassen, "Gaussian Elimination is not Optimal," *Numer. Math.* *13*, 354--356 (1969).

2. S. Winograd, "On Multiplication of 2 x 2 Matrices," *Linear Algebra and Appl.* *4*, 381--388.

3. J. Laderman, "A Noncommutative Algorithm for Multiplying 3 x 3 Matrices Using 23 Multiplications," *Bull. Amer. Math. Soc.* *82*, 126--128 (1976).

4. G. Schachtel, "A Noncommutative Algorithm for Multiplying 5 x 5 Matrices Using 103 Multiplications," *Information Processing Lett.* *7*, 180--182 (1978).

5. V. Ya. Pan, "Strassen's Algorithm is not Optimal. Trilinear Technique of Aggregating, Uniting and Canceling for Constructing Fast Algorithms for Matrix Operations," *Proc. 19th Annual Symp. on Foundations of Computer Science*, Oct. 1978, pp. 166--176.

6. S. Winograd, "Some Remarks on Fast Multiplication of Polynomials," in *Complexity of Sequential and Parallel Numerical Algorithms*, J. Traub (ed.), Academic Press, New York, 1973.